AMENDMENTS TO THE CLAIMS

Please amend the claims as follows.

1. (Currently Amended) A method for accuracy-aware analysis of a program, comprising:
   obtaining source code for the program comprising a floating-point variable;
   instrumenting the source code to associate an accuracy-aware tracking structure with the floating-point variable to obtain instrumented source code;
   compiling to instrumented source code to obtain instrumented compiled code; and
   executing the instrumented compiled code, wherein executing the instrumented compiled code comprises using the accuracy-aware tracking structure to track [[an]] a plurality of operations [[on]] applied to the floating-point variable,
   wherein data stored in the accuracy-aware tracking structure is used to determine error resulting from applying the plurality of operations and which of the plurality of operations caused the error.

2. (Original) The method of claim 1, further comprising:
   generating an accuracy-aware analysis report using the accuracy-aware tracking structure.

3. (Original) The method of claim 2, wherein the accuracy-aware analysis report includes at least one tracking variable associated with the floating-point variable selected from the group consisting of an error variable, a scaled mantissa digits variable, a renormalization variable, a left digit destruction variable, and an operations variable.

4. (Currently Amended) The method of claim 3, wherein the error variable ~~comprises~~ a half unit in last place (HULP) value associated with the floating-point variable, wherein the HULP value is a base of a floating-point representation raised to a power of the number of bits causing the error in the floating-point variable.

5. (Currently Amended) The method of claim 4, wherein ~~a value of the half unit in last place variable~~ the HULP value is determined using information obtained during renormalization.

6. (Currently Amended) The method of claim 3, wherein the error variable comprises an upper limit interval variable [[and]] or a lower limit interval variable.

7. (Original) The method of claim 3, wherein an operations variable comprises at least one selected from the group consisting of a multiplication variable, a division variable, and a square root variable.

8. (Original) The method of claim 3, wherein the renormalization variable tracks the number of addition and subtraction operations performed on the floating-point variable that do not involve left digit destruction.

9. (Currently Amended) The method of claim 1, wherein executing the compiled instrumented code comprises:

   performing [[the]] one of the plurality of operations on the floating-point variable to obtain a result;

   incrementing a tracking variable corresponding to the one of the plurality of operations associated with the floating-point variable;

   determining whether the result is exact using a scaled mantissa of the result; and

   quantifying the error associated with the result if the result is not exact.

10. (Original) The method of claim 9, further comprising:

    updating error variable using data obtained from quantifying the error associated with the result, if the result is not exact.

11. (Original) The method of claim 9, further comprising:

    determining whether the result exceeds an accuracy threshold if the result is not exact.

12. (Original) The method of claim 11, wherein execution of the compiled instrumented code halts if the accuracy threshold hold is exceeded.

13. (Original) The method of claim 11, wherein the accuracy threshold comprises at least one selected from the group consisting of a relative error threshold, an absolute error threshold, and a comparison test.

14. (Original) The method of claim 1, further comprising:
    setting an accuracy threshold for the program.

15. (Original) The method of claim 1, wherein instrumenting the source code comprises:
    parsing the source code to obtain the floating-point variable; and
    inserting additional source code to update the accuracy-aware tracking structure associated with the floating-point variable.

16. (Original) The method of claim 15, wherein the additional source code comprises functionality to call into a runtime logging utility, wherein the runtime logging utility updates the accuracy-aware tracking structure associated with the floating-point variable.

17. (Original) The method of claim 1, wherein the floating-point variable is double type.

18. – 29. (Cancelled)

30. (Original) A computer system for performing accuracy-aware analysis on a program, comprising:

a processor;

a memory;

a storage device; and

software instructions stored in the memory for enabling the computer system under control of the processor, to:

obtain source code for the program comprising a floating point variable;

instrument the source code to associate an accuracy-aware tracking structure with the floating-point variable to obtain instrumented source code;

compile to instrumented source code to obtain instrumented compiled code; and

execute the instrumented compiled code, wherein executing the instrumented compiled code comprises using the accuracy-aware tracking structure to track [[an]] a plurality of operations [[on]] applied to the floating-point variable,

wherein data stored in the accuracy-aware tracking structure is used to determine error resulting from applying the plurality of operations and which of the plurality of operations caused the error.

31. (New) The computer system of claim 30, further comprising software instructions to:

generate an accuracy-aware analysis report using the accuracy-aware tracking structure.

32. (New) The computer system of claim 31, wherein the accuracy-aware analysis report includes at least one tracking variable associated with the floating-point variable selected from the group consisting of an error variable, a scaled mantissa digits variable, a renormalization variable, a left digit destruction variable, and an operations variable.

33. (New) The computer system of claim 32, wherein the error variable a half unit in last place (HULP) value associated with the floating-point variable, wherein the HULP value is a base of a floating-point representation raised to a power of the number of bits causing the error in the floating-point variable.

34. (New) The computer system of claim 33, wherein the HULP value is determined using information obtained during renormalization.

35. (New) The computer system of claim 32, wherein an operations variable comprises at least one selected from the group consisting of a multiplication variable, a division variable, and a square root variable.

36. (New) The computer system of claim 30, wherein software instructions to execute the compiled instrumented code comprises software instructions to:

    perform one of the plurality of operations on the floating-point variable to obtain a result;

    increment a tracking variable corresponding to the one of the plurality of operations associated with the floating-point variable;

    determine whether the result is exact using a scaled mantissa of the result; and

    quantify the error associated with the result if the result is not exact.

37. (New) The computer system of claim 30, wherein software instructions for instrumenting the source code comprises software instructions to:

    parse the source code to obtain the floating-point variable; and

    insert additional source code to update the accuracy-aware tracking structure associated with the floating-point variable.